Simulation d’un déplacement de foule

Projet Java Licence 3 MIAGE

24/04/2008

BENAYOUN VINCENT
CHECCONI MAXIME
GIRAUD JULIEN
NZAOU-BILONGO PATRICK

Table des matières

1 Introduction

Dans ce document, nous allons tâcher de rendre compte du travail effectué sur le projet en répondant aux différentes questions qui nous ont été posée. Nous allons aussi montrer de quelle manière ont été traitées les différentes contraintes qui nous ont été imposées pour la réalisation de ce projet.

2 Diagramme de classes

Ce diagramme représente l'ensemble des classes que nous avons crées. Nous pouvons voir au travers de ce diagramme les interactions entre les différentes classes qui composent ce projet.

Conception des Classes

Pour la réalisation de ce projet nous avons commencé par implémenter les classes de plus haut niveau, comme Personne et ses extensions à savoir PersonneEstOuest et PersonneOuestEst, ainsi que Trottoir.

Une fois défini la manière dont nous allions modéliser notre trottoir, nous avons mis en place les méthodes permettant le déplacement d'une personne selon sa direction (nous somme parti du principe que l'on distinguait les personnes venant de l'est de celles venant de l'ouest) et le déplacement qu'elles doivent effectuer, horizontal ou nord/sud en cas de conflit avec une autre personne.

Nous avons ensuite mis en place une partie de l'interface graphique : nous avons seulement implémenté le strict nécessaire au niveau des méthodes (seules les méthodes de déplacement ont été privilégiées en premier lieu) afin de pouvoir tester notre code au fur et à mesure de son avancement. Nous avons choisi ce mode opératoire car d'une part c'est la méthode qui nous paraissait la plus sure pour mener notre projet à terme, d'autre part si nous avions implémenté l'interface une fois le code fini, nous n'aurions peut-être pas été en mesure

de faire la liaison entre l'interface graphique et notre code. De cette manière nous écartons ce risque.
De plus tester le code par le biais de l'interface met tout de suite en évidence les erreurs commises et les bugs.
Une fois cela terminé nous avons inséré les tests de conflit imposés dans le sujet. Pour le conflit de type 2 nous avons eu besoin d'implémenter la classe CelluleConflit qui permet de gérer les cellules convoitées par plusieurs personnes. Nous avons ensuite géré les différents conflits.
Une fois ce corps du projet terminé nous avons complété l'interface graphique afin qu'elle soit plus ergonomique et conviviale. Puis nous avons implémenté toute la partie concernant l'écriture des résultats statistiques dans un fichier.

Description des classes

Cellule

Concrètement, le trottoir est représenté par une matrice. Chaque case de cette matrice est une cellule. Une cellule nous donne des informations relatives aux conflits entre personnes et aux données statistiques à traiter. On peut savoir si une cellule est occupée par une personne, si celle-ci arrive de l'est ou de l'ouest.

CelluleConflit

Cette classe a été créée afin de gérer le conflit 2, c'est-à-dire lorsque plusieurs personnes désirent accéder à la même case ou cellule du trottoir.
Cette classe permet de lister toutes les personnes en conflit dans ce cas là.

Coordonnees

Comme son nom l'indique, cette classe nous fournit des coordonnées. A chaque personne marchant sur le trottoir est associé une position de coordonnées (x,y).
Le déplacement d'une personne n'est en réalité qu'un simple changement de coordonnées.

Dessin

Cette classe conserve les dimensions de la zone de dessin.

Fenêtre

C'est une des classes permettant de gérer l'interface graphique. Cette classe permet de définir les dimensions de la grille grâce à des champs longueur et largeur.

Notre trottoir état limité à une certaine dimension, on gère ici le fait que l'utilisateur désire dessiner un trottoir dont les dimensions dépassent les limites fixées.

Grille

Cette classe permet de représenter graphiquement le trottoir par une grille.

GUI

Nous avons fait appel dans cette méthode à quelques notions de programmation évènementielle.

Personne

Représente une personne sur le trottoir. Cette personne est définie par des coordonnées, une fréquence (permettant de calculer sa vitesse de déplacement), un champ de vision.

Afin de gérer au mieux les conflits, nous avons dû distinguer deux types de personnes : celles venant de l'ouest et se déplaçant vers l'est, et celles venant de l'est et se déplaçant vers l'ouest. Par conséquent, les deux classes ci-dessous héritent de la classe Personne.

PersonneEstOuest

Cette classe implémente toutes les méthodes inhérentes à une personne allant d'est en ouest. Cette classe implémente le déplacement d'une personne (horizontalement et verticalement en cas de conflit), le fait que cette personne détecte une autre personne sur son chemin (on fait la distinction entre une personne arrivant de la direction opposée et une personne prenant la même direction), la détection d'un conflit, ainsi que le calcul de la probabilité de déplacement.

PersonneOuestEst

Cette classe est quasiment similaire à la classe ci-dessus à quelques détails près, étant donné que l'on traite le cas d'une personne qui cette fois-ci se déplace d'ouest en est.

TimerBis

Cette classe implémente un timer qui permet de gérer l'insertion des personnes sur le trottoir lorsqu'on utilise l'insertion automatique.

Trottoir

La classe Trottoir nous permet d'insérer une personne sur le trottoir qui est dans le cas présent représenté par une matrice de cellules. Grâce aux méthodes implémentées dans cette classe, il nous est possible de collecter différentes informations (utilisables lors de calculs statistiques) au cours d'une simulation. Ces informations sont écrites dans un fichier comme demandé dans le sujet.

3 Mode d’emploi de l’interface graphique

Une fois le .jar exécuté, notre application lance une fenêtre d'initialisation des dimensions du trottoir, permettant de rentrer le nombre de colonnes et le nombre de lignes de la grille. Les dimensions doivent être comprises entre 10 et 35(bornes comprises), sinon un message d'erreur s'affichera indiquant que les dimensions sont erronées.

Apres avoir validé les dimensions en cliquant sur « OK », la fenêtre de la simulation de mouvement foule s'ouvre.
Cette fenêtre est composée de :

- la grille qui représente le trottoir.
- deux champs :
  - un premier permettant de rentrer le nombre de personnes que l'on souhaite insérer sur le trottoir
  - le second qui permet de rentrer la fréquence des personnes qui vont être insérées sur le trottoir (cela ne modifie pas la fréquence en temps réel, cela permet d'indiquer la fréquence pour les futurs personnes insérés).

Par default, si ces deux champs sont vides, la valeur de fréquence et le nombre de personnes sont tirées aléatoirement (entre 1 et 10 pour le nombre de personnes, et entre 1 et 5 pour la fréquence).

- différents boutons permettant le contrôle de la simulation.

Les différents boutons

Insérer : insère des personnes sur le trottoir en tenant compte des valeurs entrées pour la fréquence et le nombre de personnes désiré.
Insérer-Auto : insère des personnes sur le trottoir de façon régulière(toute les secondes).
Start : lance le timer de la simulation, ce qui permet d'exécuter la méthode actionPerformed
a chaque coup d'horloge, ce qui a pour effet de faire évoluer la simulation (mouvement des personnes...).
Stop : arrête le timer, ce qui met en pause la simulation qui pourra reprendre ultérieurement.

Enregistrer : permet d'écrire les statistiques dans un fichier nommée « statistiques », ce bouton peut être activé à tout moment de la simulation. (pas besoin de quitter le programme pour que cela écrive dans le fichier).

Redimensionner : permet de redimensionner le trottoir. En cliquant sur ce bouton, cela ferme la fenêtre de la simulation afin de rouvrir celle de l'initialisation du trottoir et de rentrer à nouveaux les dimensions souhaitées.

Clear : permet d'effacer tous ce qui est affiché à l'écran et donc de réinitialiser toutes les variables utilisées pour le calcul des statistiques. Remet également toutes les cases de notre échiquier (trottoir) à leur état initial.

Exit : permet de quitter l'application

4 Analyse du code

Méthode la plus executée

La classe contenant la méthode la plus exécutée est Personne car contenant la méthode seDeplacer() qui appelle selon le type de personne (estOuest ou ouestEst) la méthode seDeplacer() adaptée.

Elle est appelée deux fois pour chaque personne de la simulation dont l'attente est égale à leur fréquence et ce à chaque itération (ou coup d'horloge).
La première fois sert à prévoir dans le testConflit2() (voir indication supplémentaires) où va se trouver la personne à l'état futur afin de déterminer les conflits éventuels entre plusieurs personnes désirant se rendre sur une même case.
La deuxième fois sert à déplacer les personnes dans la simulation.

La classe la plus complexe

La classe qui a été la plus complexe à programmer sont celles des Personnes estOuest/ouestEst (identiques à peu de choses) car c'est à l'intérieur de ces classes que se trouvent le code gérant les différents types conflits, les déplacement horizontaux, verticaux et les blocages avec tous les différents cas à traiter que cela implique.
Il s'agit ici de prévoir toutes les différentes situations dans lesquelles pourraient se retrouver un individu inséré dans la simulation.

Le nombre d'appel à random ?

Lorsque l'on veut créer des personnes à partir de (l'interface graphique) on doit rentrer une fréquence et un nombre de personnes, si les champs sont laissés à vide 2 nombres aléatoires sont générés.

Lors de l'appel à la fonction d'insertion d'une personne 2 autres nombres aléatoires sont tirés, un pour la direction et l'autre pour sa position verticale, celui peut être tiré jusqu'à 4 fois (si on ne peut insérer une personne à une case on tire un autre nombre aléatoire, on répète cette opération jusqu'à 4 fois sinon la personne ne sera pas insérée)

Enfin lorsque une personne en aperçoit une autre en sens inverse (ce tirage s'effectue donc pour les 2 personnes), on tire un premier nombre aléatoire pour savoir si elle doit ce déplacer, puis un deuxième pour connaitre son type de déplacement.

Indications supplémentaires

La gestion des conflit de type 2 (2ème exemple du sujet) : testConflit2() à été implémentée dans la classe Trottoir mais cette dernière causant des problèmes que nous n'avons pas réussi à corriger durant le développement de notre programme nous avons décidé de ne pas la lancer lors de la simulation c'est pourquoi elle se trouve en commentaire dans la méthode actualise() de la classe Trottoir (testConflit2()).

Elle provoquait la disparition aléatoire d'individus de la simulation en laissant la case de leur dernière position comme étant occupée, provoquant lors du passage d'autres individus des blocages.

5 Simulation

6 JavaDoc et .Jar

La Javadoc a été générée dans les fichiers du projet.
Le point .jar se trouve dans le dossier /dist.